Probabilistic Model Checking on Propositional Projection Temporal Logic [⋆]

Xiaoxiao Yang

State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences
Beijing, 100190, China
xxyang@ios.ac.cn

Abstract. Propositional Projection Temporal Logic (PPTL) is a useful formalism for reasoning about period of time in hardware and software systems and can handle both sequential and parallel compositions. In this paper, based on discrete time Markov chains, we investigate the probabilistic model checking approach for PPTL towards verifying arbitrary linear-time properties. We first define a normal form graph, denoted by NFG_{inf}, to capture the infinite paths of PPTL formulas. Then we present an algorithm to generate the NFG_{inf}. Since discrete-time Markov chains are the deterministic probabilistic models, we further give an algorithm to determinize and minimize the nondeterministic NFG_{inf} following the Safra's construction.

Keywords: projection temporal logic, probabilistic model checking, Markov chains, normal form graph.

1 Introduction

Traditional model checking techniques focus on a systematic check of the validity of a temporal logic formula on a precise mathematical model. The answer to the model checking question is either true or false. Although this classic approach is enough to specify and verify boolean temporal properties, it does not allow to reason about stochastic nature of systems. In real-life systems, there are many phenomena that can only be modeled by considering their stochastic characteristics. For this purpose, probabilistic model checking is proposed as a formal verification technique for the analysis of stochastic systems. In order to model random phenomena, discrete-time Markov chains, continuous-time Markov chains and Markov decision processes are widely used in probabilistic model checking.

Linear-time property is a set of infinite paths. We can use linear-time temporal logic (LTL) to express ω-regular properties. Given a finite Markov chain M and an ω-regular property Q, the probabilistic model checking problem for LTL

[⋆] This research is supported by the NSFC No.60721061, China Postdoctoral Science Foundation and the CAS Innovation Programs.

is to compute the probability of accepting runs in the product Markov chain M and a deterministic Rabin automata (DRA) for $\neg Q$ [6].

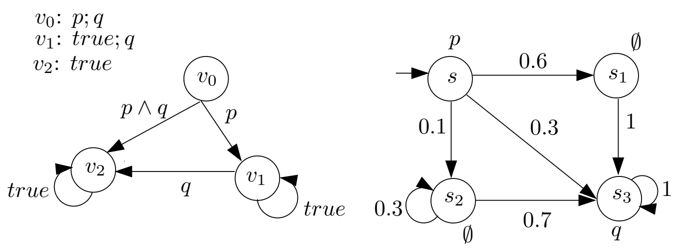
Among linear-time temporal logics, there exists a number of *choppy logics* that are based on chop (**;**) operators. Interval Temporal Logic (ITL) [3] is one kind of choppy logics, in which temporal operators such as *chop*, *next* and *projection* are defined. Within the ITL developments, Duan, Koutny and Holt, by introducing a new projection construct $(p_1, \ldots, p_m)\, prj\ q$, generalize ITL to infinite time intervals. The new interval-based temporal logic is called Projection Temporal Logic (PTL) [12]. PTL is a useful formalism for reasoning about period of time for hardware and software systems. It can handle both sequential and parallel compositions, and offer useful and practical proof techniques for verifying concurrent systems [14,12]. Compared with LTL, PTL can describe more linear-time properties. In this paper, we investigate the probabilistic model checking on Propositional PTL (PPTL).

There are a number of reasons for being interested in projection temporal logic language. One is that projection temporal logic can express various imperative programming constructs (e.g. while-loop) and has executable subset [10,11]. In addition, the expressiveness of projection temporal logic is more powerful than the classic point-based temporal logics such as LTL since the temporal logics with *chop star* ($*$) and *projection* operators are equivalent to ω-regular languages, but LTL cannot express all ω-regular properties [9]. Furthermore, the key construct used in PTL is the new projection operator $(p_1, \ldots, p_m)\ prj\ \ q$ that can be thought of as a combination of the parallel and the projection operators in ITL. By means of the projection construct, one can define fine- and coarse-grained concurrent behaviors in a flexible and readable way. In particular, the sequence of processes $p_1, \ldots, p_m$ and process q may terminate at different time points.

In the previous work [10,11,12], we have presented a *normal form* for any PPTL formula. Based on the normal form, we can construct a semantically equivalent graph, called *normal form graph* (NFG). An infinite (finite) interval that satisfies a PPTL formula will correspond to an infinite (finite) path in NFG. Different from Buchi automata, NFG is exactly the model of a PPTL formula. For any unsatisfiable PPTL formula, NFG will be reduced to a false node at the end of the construction. NFG consists of both finite and infinite paths. But for concurrent stochastic systems, here we only consider infinite cases. Therefore, we define NFG_{inf} to denote an NFG only with infinite paths. To capture the accurate semantics for PPTL formulas with infinite intervals, we adopt Rabin acceptance condition as accepting states in NFG_{inf}. In addition, since Markov chain M is a deterministic probabilistic model, in order to guarantee that the product of $M \otimes NFG_{inf}$ is also a Markov chain, we give an algorithm for deterministic NFG_{inf}, in the spirit of Safra's construction for deterministic Buchi-automata.

To make this idea clear, we now consider a simple example shown in Figure 1. The definitions of NFGs and Markov chains are formalized in the subsequent sections. Let $p\ ;\ q$ be a *chop* formula in PPTL, where p and q are atomic propositions. NFG_{inf} of $p\ ;\ q$ is constructed in Figure 1(a), where nodes v_0, v_1 and v_2

are temporal formulas, and edges are state formulas (without temporal operators). v_0 is an initial node. v_2 is an acceptance node recurring for infinitely many times, whereas v_1 appears finitely many times. Figure 1(b) presents a Markov chain with initial state s. Let path $path = \langle s, s_1, s_3 \rangle$. We can see that $path$ satisfies $p\ ;\ q$ with probability 0.6. Based on the product of Markov chain and NFG_{inf}, we can compute the whole probability that the Markov chain satisfies $p\ ;\ q$.



(a) NFG_{inf} of $p;q$. (b) An Example of Markov chains.

Fig. 1. A Simple Example for Probabilistic Model Checking on PPTL.

Compared with Buchi automata, NFGs have the following advantages that are more suitable for verification for interval-based temporal logics.
(i) NFGs are beneficial for unified verification approaches based on the same formal notation. NFGs can not only be regarded as models of specification language PTL, but also as models of Modeling Simulation and Verification Language (MSVL)[10,11], which is an executable subset of PTL. Thus, programs and their properties can be written in the same language, which avoids the transformation between different notations.
(ii) NFGs can accept both finite words and infinite words. But Buchi automata can only accept infinite words. Further, temporal operators *chop* ($p\ ;\ q$), *chop star* (p^*), and *projection* can be readily transformed to NFGs.
(iii) NFGs and PPTL formulas are semantically equivalent. That is, every path in NFGs corresponds to a model of PPTL formula. If some formula is false, then its NFG will be a false node. Thus, satisfiability in PPTL formulas can be reduced to NFGs construction. But for any LTL formula, the satisfiability problem needs to check the emptiness problem of Buchi automata.

The paper is organized as follows. Section 2 introduces PPTL briefly. Section 3 presents the (discrete time) Markov chains. In Section 4, the probabilistic model checking approach for PPTL is investigated. Finally, conclusions are drawn in Section 5.

2 Propositional Projection Temporal Logic

The underlying logic we use is Propositional Projection Temporal Logic (PPTL). It is a variation of Propositional Interval Temporal Logic (PITL).

Definition 1 Let AP be a finite set of atomic propositions. PPTL formulas over AP can be defined as follows:

$$Q ::= \pi \mid \neg Q \mid \bigcirc Q \mid Q_1 \wedge Q_2 \mid (Q_1, \ldots, Q_m)\, prj\ Q \mid Q^+$$

where $\pi \in AP$, $Q, Q_1, \ldots, Q_n$ are PPTL formulas, $\bigcirc$ (next), prj (projection) and $+$ (plus) are basic temporal operators.

A formula is called a *state* formula if it does not contain any temporal operators, i.e., *next* ($\bigcirc$), *projection* (prj) and *chop-plus* ($^+$); otherwise it is a *temporal* formula.

An interval $\sigma = \langle s_0, s_1, \ldots \rangle$ is a non-empty sequence of states, where s_i $(i \geq 0)$ is a state mapping from AP to $B = \{true, false\}$. The length, $|\sigma|$, of σ is ω if σ is infinite, and the number of states minus 1 if σ is finite. To have a uniform notation for both finite and infinite intervals, we will use *extended integers* as indices. That is, for set N_0 of non-negative integer and ω, we define $N_\omega = N_0 \cup \{\omega\}$, and extend the comparison operators: $=, <, \leq$, to N_ω by considering $\omega = \omega$, and for all $i \in N_0, i < \omega$. Moreover, we define $\preceq$ as $\leq -\{(\omega, \omega)\}$.

To define the semantics of the projection construct we need an auxiliary operator. Let $\sigma = \langle s_0, s_1, ... \rangle$ be an interval and $r_1, \ldots, r_h$ be integers $(h \geq 1)$ such that $0 \leq r_1 \leq \ldots \leq r_h \preceq |\sigma|$.

$$\sigma \downarrow (r_1, \ldots, r_h) \overset{\text{def}}{=} \langle s_{t_1}, s_{t_2}, \ldots, s_{t_l} \rangle$$

The *projection* of σ onto $r_1, \ldots, r_h$ is the interval (called projected interval) where $t_1, \ldots, t_l$ are obtained from $r_1, \ldots, r_h$ by deleting all duplicates. In other words, $t_1, \ldots, t_l$ is the longest strictly increasing subsequence of $r_1, \ldots, r_h$. For example, $\langle s_0, s_1, s_2, s_3 \rangle \downarrow (0, 2, 2, 2, 3) = \langle s_0, s_2, s_3 \rangle$. As depicted in Figure 2, the projected interval $\langle s_0, s_2, s_3 \rangle$ can be obtained by using $\downarrow$ operator to take the endpoints of each process $\varepsilon, len(2), \varepsilon, \varepsilon, len(1)$.

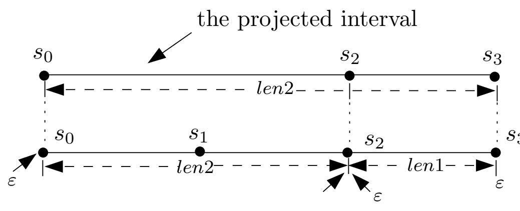


Fig. 2. A projected interval.

An interpretation for a PPTL formula is a tuple $\mathcal{I} = (\sigma, i, k, j)$, where σ is an interval, i, k are integers, and j an integer or ω such that $i \leq k \preceq j$. Intuitively,

(σ, i, k, j) means that a formula is interpreted over a subinterval $\sigma_{(i,..,j)}$ with the current state being s_k. The satisfaction relation ($\models$) between interpretation $\mathcal{I}$ and formula Q is inductively defined as follows.

1. $\mathcal{I} \models \pi$ iff $s_k[\pi] = true$
2. $\mathcal{I} \models \neg Q$ iff $\mathcal{I} \not\models Q$
3. $\mathcal{I} \models Q_1 \wedge Q_2$ iff $\mathcal{I} \models Q_1$ and $\mathcal{I} \models Q_2$
4. $\mathcal{I} \models \bigcirc Q$ iff $k < j$ and $(\sigma, i, k+1, j) \models Q$
5. $\mathcal{I} \models (Q_1, \ldots, Q_m)\ prj\ Q$ iff there are $k = r_0 \leq r_1 \leq \ldots \leq r_m \preceq j$ such that $(\sigma, i, r_0, r_1) \models Q_1$ and $(\sigma, r_{l-1}, r_{l-1}, r_l) \models Q_l$ for all $1 < l \leq m$ and $(\sigma', 0, 0, |\sigma'|)$ $\models Q$ for σ' given by :
   (a) $r_m < j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_m) \cdot \sigma_{(r_m+1,..,j)}$
   (b) $r_m = j$ and $\sigma' = \sigma \downarrow (r_0, \ldots, r_h)$ for some $0 \leq h \leq m$.
6. $\mathcal{I} \models Q^+$ iff there are finitely many $r_0, \ldots, r_n$ and $k = r_0 \leq r_1 \leq \ldots \leq r_{n-1} \preceq r_n = j$ $(n \geq 1)$
   such that $(\sigma, i, r_0, r_1) \models Q$ and $(\sigma, r_{l-1}, r_{l-1}, r_l) \models Q$ for all $1 < l \leq n$ or $j = \omega$ and there are infinitely many integers $k = r_0 \leq r_1 \leq r_2 \leq \ldots$ such that $\lim\limits_{i \to \infty} r_i = \omega$ and $(\sigma, i, r_0, r_1) \models Q$ and for $l > 1, (\sigma, r_{l-1}, r_{l-1}, r_l) \models Q$.

A PPTL formula Q is satisfied by an interval σ, denoted by $\sigma \models Q$, if $(\sigma, 0, 0, |\sigma|) \models Q$. A formula Q is called satisfiable, if $\sigma \models Q$. A formula Q is valid, denoted by $\models Q$, if $\sigma \models Q$ for all σ. Sometimes, we denote $\models p \leftrightarrow q$ (resp. $\models p \rightarrow q$) by $p \approx q$ (resp.$\hookrightarrow$) and $\models \Box(p \leftrightarrow q)$ (resp. $\models \Box(p \rightarrow q)$) by $p \equiv q$ (resp. $p \supset q$), The former is called *weak equivalence (resp. weak implication)* and the latter *strong equivalence (resp. strong implication).*

Figure 3 below shows us some useful formulas derived from elementary PTL formulas. ε represents the final state and *more* specifies that the current state is a non-final state; $\Diamond P$ (namely *sometimes* P) means that P holds eventually in the future including the current state; $\Box P$ (namely *always* P) represents that P holds always in the future from now on; $\odot P$ (*weak next*) tells us that either the current state is the final one or P holds at the next state of the present interval; $Prj(P_1, \ldots, P_m)$ represents a *sequential* computation of $P_1, \ldots, P_m$ since the projected interval is a singleton; and $P\ ;\ Q$ (P *chop* Q) represents a computation of P followed by Q, and the intervals for P and Q share a common state. That is, P holds from now until some point in future and from that time point Q holds. Note that $P\ ;\ Q$ is a strong chop which always requires that P be true on some finite subinterval. $len(n)$ specifies the distance n from the current state to the final state of an interval; *skip* means that the length of the interval is one unit of time. *fin*(P) is *true* as long as P is *true* at the final state while *keep*(P) is *true* if P is true at every state but the final one. The formula *halt*(P) holds if and only if formula P is *true* at the final state.

An Application of Projection Construct

Example 1 We present a simple application of projection construct about a pulse generator for variable x which can assume two values: 0 (low) and 1 (high).

$$\begin{array}{ll}
\varepsilon & \overset{\text{def}}{=} \neg \bigcirc \mathit{true} \\
\mathit{len}(n) & \overset{\text{def}}{=} \begin{cases} \varepsilon & \text{if } n = 0 \\ \bigcirc \mathit{len}(n-1) & \text{if } n > 1 \end{cases} \\
\Box P & \overset{\text{def}}{=} \neg\Diamond\neg P \\
\mathit{skip} & \overset{\text{def}}{=} \mathit{len}(1) \\
\mathit{Prj}(P_1, \ldots, P_m) & \overset{\text{def}}{=} (P_1, \ldots, P_m)\ \mathit{prj}\ \varepsilon \\
\mathit{fin}(P) & \overset{\text{def}}{=} \Box(\varepsilon \to P) \\
P \,;\, Q & \overset{\text{def}}{=} \mathit{Prj}\ (P, Q) \\
\mathit{keep}(P) & \overset{\text{def}}{=} \Box(\neg\varepsilon \to P) \\
\mathit{more} & \overset{\text{def}}{=} \neg\varepsilon \\
\mathit{halt}(P) & \overset{\text{def}}{=} \Box(\varepsilon \leftrightarrow P) \\
\Diamond P & \overset{\text{def}}{=} \mathit{Prj}(\mathit{true}, P) \\
\odot P & \overset{\text{def}}{=} \varepsilon \vee \bigcirc P
\end{array}$$

Fig. 3. Derived PPTL formulas.

We first define two types of processes: The first one is $\mathit{hold}(i)$ which is executed over an interval of length i and ensures that the value of x remains constant in all but the final state,

$$\mathit{hold}(i) \overset{\text{def}}{=} \mathit{frame}(i) \wedge \mathit{len}(i)$$

The other is $\mathit{switch}(j)$ which is ensures that the value of x is first set to 0 and then changed at every subsequent state,

$$\mathit{switch}(j) \overset{\text{def}}{=} x = 0 \wedge \mathit{len}(j) \wedge \Box(\mathit{more} \to \bigcirc x = 1 - x)$$

Having defined $\mathit{hold}(i)$ and $\mathit{switch}(j)$, we can define the pulse generators with varying numbers and length of low and high intervals for x,

$$\mathit{pulse}(i_1, \ldots, i_k) \overset{\text{def}}{=} (\mathit{hold}(i_1), \ldots, \mathit{hold}(i_k))\ \mathit{prj}\ \mathit{switch}(k)$$

For instance, a pulse generator

$$\mathit{pulse}(3, 5, 3, 4) \overset{\text{def}}{=} (\mathit{hold}(3), \mathit{hold}(5), \mathit{hold}(3), \mathit{hold}(4))\ \mathit{prj}\ \mathit{switch}(4)$$

can be shown in Figure 4.

Let Q be a PPTL formula and $Q_p \in AP$ be a set of atomic propositions in Q. Normal form of PPTL formulas can be defined as follows.

Definition 2 A PPTL formula Q is in *normal form* if

$$Q \equiv (\bigvee_{j=0}^{n_0} Q_{e_j} \wedge \varepsilon) \vee (\bigvee_{i=0}^{n} Q_{c_i} \wedge \bigcirc Q_{f_i})$$

```
  |<-------------------------   switch(4)     ------------------------------>|

x=0              1                        0              1                   0
  |--------------|------------------------|--------------|-------------------|
  t0             t3                       t8             t11                 t15

  t0   t1   t2   t3   t4   t5   t6   t7   t8   t9   t10  t11  t12  t13  t14  t15
  |----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
  |<--hold(3)--->|<-------hold(5) ------->|<--hold(3)--->|<---- hold(4) ---->|
x=0    0    0    1    1    1    1    1    0    0    0    1    1    1    1    0
```

Fig. 4. A Pulse Generator

where $Q_{e_j} \equiv \bigwedge_{k=1}^{m_0} \dot{q_{jk}}, Q_{c_i} \equiv \bigwedge_{h=1}^{m} \dot{q_{ih}}$, $|Q_p| = l$, $1 \leq m_0 \leq l$, $1 \leq m \leq l$; $q_{jk}, q_{ih} \in Q_p$, for any $r \in Q_p$, $\dot{r}$ means r or $\neg r$; Q_{fi} is a general PPTL formula. For convenience, we often write $Q_e \wedge \varepsilon$ instead of $\bigvee_{j=0}^{n_0} Q_{e_j} \wedge \varepsilon$ and $\bigvee_{i=0}^{n} Q_i \wedge \bigcirc Q'_i$ instead of $\bigvee_{i=0}^{n} Q_{c_i} \wedge \bigcirc Q_{f_i}$. Thus,

$$Q \equiv (Q_e \wedge \varepsilon) \vee (\bigvee_{i=0}^{n} Q_i \wedge \bigcirc Q'_i)$$

where Q_e and Q_i are state formulas.

Theorem 1 For any PPTL formula Q, there is a normal form Q' such that $Q \equiv Q'$. [12]

3 Probabilistic System

We model probabilistic system by *(discrete-time) Markov chains* (DTMC). Without loss of generality, we assume that a DTMC has a unique initial state.

Definition 3 A Markov chain is a tuple $M = (S, Prob, \iota_{init}, AP, L)$, where S is a countable, nonempty set of states; $Prob : S \times S \to [0, 1]$ is the transition probability function such that $\sum_{s' \in S} Prob(s, s') = 1$; $\iota_{init} : S \to [0, 1]$ is the initial distribution such that $\sum_{s \in S} \iota_{init}(s) = 1$, and AP is a set of atomic propositions and $L : S \to 2^{AP}$ a labeling function.

As in the standard theory of Markov processes [8], we need to formalize a probability space of M that can be defined as $\psi_M = (\Omega, Cyl, Pr)$, where Ω denotes the set of all infinite sequences of states $\langle s_0, s_1, \ldots \rangle$ such that $Prob(s_i, s_{i+1}) >$

0 for all $i \leq 0$, Cyl is a σ-algebra generated by the *basic cylindric sets*:

$$Cyl(s_0, \ldots, s_n) = \{path \in \Omega \mid path = s_0, s_1, \ldots, s_n, \ldots\}$$

and Pr is a probability distribution defined by

$$\begin{aligned} Pr^M(Cyl(s_0, \ldots, s_n)) &= Prob(s_0, \ldots, s_n) \\ &= \prod_{0 \leq i < n} Prob(s_i, s_{i+1}) \end{aligned}$$

If p is a path in DTMC M and Q a PPTL formula, we often write $p \models Q$ to mean that a path in DTMC satisfies the given formula Q. Let $path(s)$ be a set of paths in DTMC starting with state s. The probability for Q to hold in state s is denoted by $Pr^M(s \models Q)$, where $Pr^M(s \models Q) = Pr^M_s\{p \in path(s) \mid p \models Q\}$.

4 Probabilistic Model Checking for PPTL

In [12], it is shown that any PPTL formulas can be rewritten into normal form, where a graphic description for normal form called Normal Form Graph (NFG) is presented. NFG is an important basis of decision procedure for satisfiability and model checking for PPTL. In this paper, the work reported depends on the NFG to investigate the probabilistic model checking for PPTL.

However, there are some differences on NFG between our work and the previous work in [10,11,12]. First, NFG consists of finite paths and infinite paths. For concurrent stochastic systems, we only consider to verify ω-regular properties. Thus, we are supposed to concern with all the infinite paths of NFG. These infinite paths are denoted by NFG_{inf}. Further, to define the nodes which recur for finitely many times, [12] uses Labeled NFG (LNFG) to tag all the nodes in finite cycles with F. But it can not identify all the possible acceptance cases. As the standard acceptance conditions in ω-automata, we adopt Rabin acceptance condition to precisely define the infinite paths in NFG_{inf}. In addition, since Markov chain M is a deterministic probabilistic model, in order to guarantee that the product of $M \otimes NFG_{inf}$ is also a Markov chain, the NFG_{inf} needs to be deterministic. Thus, following the Safra's construction for deterministic automata, we design an algorithm to obtain a deterministic NFG_{inf}.

4.1 Normal Form Graph

In the following, we first give a general definition of NFG for PPTL formulas.

Definition 4 (Normal Form Graph [10,12]) For a PPTL formula P, the set $V(P)$ of nodes and the set of $E(P)$ of edges connecting nodes in $V(P)$ are inductively defined as follows.

1. $P \in V(P)$;

2. For all $Q \in V(P)/\{\varepsilon, \mathit{false}\}$, if $Q \equiv (Q_e \wedge \varepsilon) \vee (\bigvee_{i=0}^{n} Q_i \wedge \bigcirc Q_i')$, then $\varepsilon \in V(P)$, $(Q, Q_e, \varepsilon) \in E(P)$; $Q_i' \in V(P)$, $(Q, Q_i, Q_i') \in E(P)$ for all i, $1 \leq i \leq n$.

The NFG of PPTL formula P is the directed graph $G = (V(P), E(P))$.

A finite path for formula Q in NFG is a sequence of nodes and edges from the root to node ε. while an infinite path is an infinite sequence of nodes and edges originating from the root.

Theorem 2 (Finiteness of NFG) For any PPTL formula P, $|V(P)|$ is finite [12].

Theorem 2 assures that the number of nodes in NFG is finite. Thus, each satisfiable formula of PPTL is satisfiable by a finite transition system (i.e., finite NFG). Further, by the finite model property, the satisfiability of PPTL is decidable. In [12], Duan *etal* have given a decision procedure for PPTL formulas based on NFG.

To verify ω-regular properties, we need to consider the infinite paths in NFG. By ignoring all the finite paths, we can obtain a subgraph only with infinite paths, denoted $\mathit{NFG}_{\mathit{inf}}$.

Definition 5 For a PPTL formula P, the set $V_{\mathit{inf}}(P)$ of nodes and the set of $E_{\mathit{inf}}(P)$ of edges connecting nodes in $V_{\mathit{inf}}(P)$ are inductively defined as follows.

1. $P \in V_{\mathit{inf}}(P)$;
2. For all $Q \in V_{\mathit{inf}}(P)$, if $Q \equiv (Q_e \wedge \varepsilon) \vee (\bigvee_{i=0}^{n} Q_i \wedge \bigcirc Q_i')$, then $Q_i' \in V_{\mathit{inf}}(P)$, $(Q, Q_i, Q_i') \in E_{\mathit{inf}}(P)$ for all i, $1 \leq i \leq n$.

Thus, $\mathit{NFG}_{\mathit{inf}}$ is a directed graph $G' = (V_{\mathit{inf}}(P), E_{\mathit{inf}}(P))$. Precisely, G' is a subgraph of G by deleting all the finite path from node P to node ε.

In fact, a finite path in the NFG of a formula Q corresponds to a model (i.e., interval) of Q. However, the result does not hold for the infinite case since not all of the infinite paths in NFG can be the models of Q. Note that, in an infinite path, there must exist some nodes which appear infinitely many times, but there may have other nodes that can just recur for finitely many times. To capture the precise semantics model of formula Q, we make use of Rabin acceptance condition as the constraints for nodes that must recur finitely.

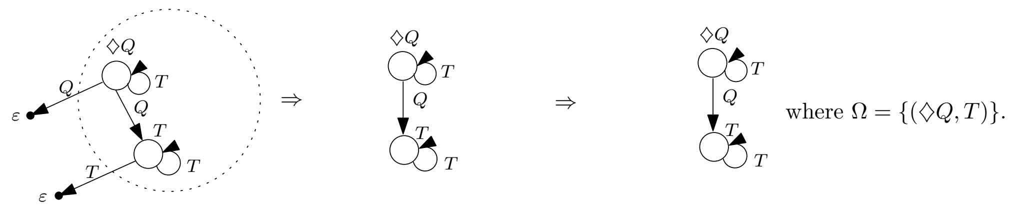
Definition 6 For a PPTL formula P, $\mathit{NFG}_{\mathit{inf}}$ with Rabin acceptance condition is defined as $G_{\mathit{Rabin}} = (V_{\mathit{inf}}(P), E_{\mathit{inf}}(P), v_0, \Omega)$, where $V(P)$ is the set of nodes and $E(P)$ is the set of directed edges between $V(P)$, $v_0 \in V(P)$ is the initial node, and $\Omega = \{(E_1, F_1), \ldots, (E_k, F_k)\}$ with $E_i, F_i \in V(P)$ is Rabin acceptance condition. We say that: an infinite path is a model of the formula P if there exists an infinite run ρ on the path such that

$$\exists (E, F) \in \Omega . (\rho \cap E = \emptyset) \wedge (\rho \cap F \neq \emptyset)$$

Example 2 Let Q be PPTL formulas. The normal form of $\Diamond Q$ are as follows.

$$\begin{aligned}
\Diamond Q &\equiv \mathit{true}\ ;\ Q\\
&\equiv (\varepsilon \vee \bigcirc \mathit{true})\ ;\ Q\\
&\equiv (\varepsilon\ ;\ Q) \vee (\bigcirc \mathit{true}\ ;\ Q)\\
&\equiv Q \vee \bigcirc(\mathit{true};Q)\\
&\equiv (Q \wedge \varepsilon) \vee (Q \wedge \bigcirc \mathit{true}) \vee \bigcirc\Diamond Q\\
&\equiv (Q \wedge \varepsilon) \vee (Q \wedge \bigcirc(\varepsilon \vee \bigcirc \mathit{true})) \vee \bigcirc\Diamond Q
\end{aligned}$$

The NFG and $\mathit{NFG}_{\mathit{inf}}$ with Rabin acceptance condition of $\Diamond Q$ are depicted in Figure 5. By the semantics of formula $\Diamond Q$ (see Figure 3), that is, formula Q holds eventually in the future including the current state, we can know that node $\Diamond Q$ must cycle for finitely many times and node T (i.e., *true*) for infinitely many times.



(i) NFG of $\Diamond Q$. (ii) $\mathit{NFG}_{\mathit{inf}}$ of $\Diamond Q$. (iii) $\mathit{NFG}_{\mathit{inf}}$ with Rabin acceptance condition of $\Diamond Q$.

Fig. 5. NFG of $\Diamond Q$.

4.2 The Algorithms

To investigate the probabilistic model checking problem for interval-based temporal logics, we use Markov chain M as stochastic models and PPTL as a specification language. In the following, we present algorithms for the construction and determinization of $\mathit{NFG}_{\mathit{inf}}$ with Rabin acceptance condition respectively.

Construction of $\boldsymbol{\mathit{NFG}_{\mathit{inf}}}$ In Table 1, we present algorithm $\mathit{NFG}_{\mathit{inf}}(Q)$ for constructing the $\mathit{NFG}_{\mathit{inf}}$ with Rabin acceptance condition for any PPTL formula. Algorithm $\mathit{NF}(Q)$ can be found in [12], which is used for the purpose of transforming formula Q into its normal form. For any formula $R \in V_{\mathit{inf}}(Q)$ and $\mathit{visit}(R) = 0$, we assume that $P = \mathit{NF}(R)$ is in normal form, where $\mathit{visit}(R) = 0$ means that formula R has not been decomposed into its normal form. When $P \equiv \bigvee_{i=1}^{k} P_i \vee \bigcirc P'_i$ or $P \equiv (\bigvee_{j=1}^{h} P_{ej} \wedge \varepsilon) \vee (\bigvee_{i=1}^{k} P_i \wedge \bigcirc P'_i)$, if P'_i is a new

Table 1. Algorithm for constructing NFG_{inf} with Rabin condition for a PPTL formula.

```
Function $NFG_{inf}(Q)$
/*precondition: Q is a PPTL formula, NF(Q) is the normal form for $Q$ */
/*postcondition: $NFG_{inf}(Q)$ outputs $NFG_{inf}$ with Rabin condition of $Q$,
                 $G_{Rabin} = (V_{inf}(Q), E_{inf}(Q), v_0, \Omega)$ */

begin function
  $V_{inf}(Q) = \{Q\}; E_{inf}(Q) = \emptyset;$ visit$(Q) = 0; v_0 = Q; E = F = \emptyset;$   /*initialization*/
   while there exists $R \in V_{inf}(Q)$ and visit(R) == 0
    do $P = NF(R)$;   visit$(R) = 1$;
    switch$(P)$
        case $P \equiv \bigvee_{j=1}^{h} P_{ej} \wedge \varepsilon$: break;
        case $P \equiv \bigvee_{i=1}^{k} P_i \wedge \bigcirc P'_i$ or $P \equiv (\bigvee_{j=1}^{h} P_{ej} \wedge \varepsilon) \vee (\bigvee_{i=1}^{k} P_i \wedge \bigcirc P'_i)$:
                foreach $i$ $(1 \leq i \leq k)$ do
                   if  $\neg(P'_i \equiv false)$  and $P'_i \notin V_{inf}(Q)$
                   then visit$(P'_i) = 0$;
                   /*$P_i$ is not decomposed to normal form*/
                        $V_{inf}(Q) = V_{inf}(Q) \cup \bigcup_{i=1}^{k} \{P'_i\}$;
                        $E_{inf}(Q) = E_{inf}(Q) \cup \bigcup_{i=1}^{k} \{(R, P_i, P'_i)\}$;
                   if $\neg(P'_i \equiv false)$  and $P'_i \in V_{inf}(Q)$
                   then $E_{inf}(Q) = E_{inf}(Q) \cup \bigcup_{i=1}^{k} \{(R, P_i, P'_i)\}$;
                        when $P'_i = R$  do  /*self-loop*/
                          if $R$ is $Q_1$ ; $Q_2$ then $E = E \cup \{R\}$ else $F = F \cup \{R\}$
                        for some node $R'' \in V_{inf}(Q)$;
                        let $NF(R'') = \bigvee_{j=1}^{k} R_i \wedge \bigcirc R$ or
                            $NF(R'') = (\bigvee_{j=1}^{h} R_{ej} \wedge \varepsilon) \vee (\bigvee_{i=1}^{k} R_i \wedge \bigcirc R)$;
                        /*nodes $R$ and $R''$ form a loop*/
                        when $P'_i = R''$ $(R'' \neq R)$ do
                          if $R, R'' \notin E$ then $F = F \cup \{\{R, R''\}\}$
                          else $E = E \cup \{\{R, R''\}\}$;
        break;
  end while
  return $G_{Rabin}$;
end function
```

formula (node), that is, $P_i' \notin V_{inf}$, then by Definition 5, we add the new node P_i' to V_{inf} and edge (R, P_i, P_i') to E_{inf} respectively. On the other hand, if $P_i' \in V_{inf}$, then it will be a loop. In particular, we need to consider the case of $R \equiv Q_1 \mathbin{;} Q_2$. Because $Q_1 \mathbin{;} Q_2$ (Q_1 *chop* Q_2, defined in Fig.3) represents a computation of Q_1 followed by Q_2, and the intervals for Q_1 and Q_2 share a common state. That is, Q_1 holds from now until some point in future and from that time point Q_2 holds. Note that $Q_1 \mathbin{;} Q_2$ used here is a *strong chop* which always requires that Q_1 be true on some finite subinterval. Therefore, infinite models of Q_1 can cause R to be false. To solve the problem, we employ Rabin acceptance condition to constraint that chop formula will not be repeated infinitely many times.

By Theorem 2, we know that nodes $V(Q)$ is finite in *NFG*. Since $V_{inf}(Q) \subseteq V(Q)$, so $V_{inf}(Q)$ is finite as well. This is essential since it can guarantee that the algorithm $NFG_{inf}(Q)$ will terminate.

Theorem 3 Algorithm $NFG_{inf}(Q)$ always terminates.

Proof: Let $V_{inf}(Q) = \{v_1, \ldots, v_n\}$. When all nodes in V_{inf} are transformed into normal form, we have $visit(v_i) == 1$ $(1 \leq i \leq n)$. Hence, the while loop always terminates.

We denote the set of infinite paths in an NFG_{inf} G by $path(G) = \{p_1, \ldots, p_m\}$, where p_i $(1 \leq i \leq m)$ is an infinite path from the initial node to some acceptable node in F. The following theorem holds.

Theorem 4 G_{Rabin} and G'_{Rabin} are equivalent if and only if $path(G_{Rabin}) = path(G'_{Rabin})$.

Let Q be a satisfiable PPTL formula. By unfolding the normal form of Q, there is a sequence of formulas $\langle Q, Q_1, Q_1', Q_2, Q_2', \ldots \rangle$. Further, by algorithm NFG_{inf}, we can obtain an equivalent NFG_{inf} to the normal form. In fact, an infinite path in NFG_{inf} of Q corresponds to a model of Q. We conclude this fact in Theorem 5.

Theorem 5 A formula Q can be satisfied by infinite models if and only if there exists infinite paths in NFG_{inf} of Q with Rabin acceptance condition.

Determinization of NFG_{inf} Buchi automata and NFG_{inf} both accept ω-words. The former is a basis for the automata-theoretic approach for model checking with liner-time temporal logic, whereas the latter is the basis for the satisfiability and model checking of PPTL formulas. Following the thought of the Safra's construction for deterministic Buchi automata [15], we can obtain a deterministic NFG_{inf} with Rabin acceptance condition from the non-deterministic ones. However, different from the states in Buchi automata, each node in NFG_{inf} is specified by a formula in PPTL. Thus, by eliminating the nodes that contain equivalent formulas, we can decrease the number of states in the resulting deterministic NFG_{inf} to some degree.

The construction for deterministic NFG_{inf} is shown in Table 2. For any $R \in V'_{inf}(Q)$, R is a Safra tree consisting of a set of nodes, and each node v is a set of formulas. By Safra's algorithm [15], we can compute all reachable Safra tree R' that can be reached from R on input P_i. To obtain a deterministic NFG_{inf}, we take all pairs (E_v, F_v) as acceptance component, where E_v consists of all Safra trees without a node v, and F_v all Safra trees with node v marked '!' that denotes v will recur infinitely often. Furthermore, we can minimize the number of states in the resulting NFG_{inf} by finding equivalent nodes. Let $R = \{v_0, \ldots, v_n\}$ and $R' = \{v'_0, \ldots, v'_n\}$ be two Safra's trees, where $R, R' \in V'_{inf}$, nodes $v_i = \{Q_1, Q_2, \ldots\}$ and $v'_i = \{Q'_1, Q'_2, \ldots\}$ be a set of formulas. For any nodes v_i and v'_i, if we have $v_i = v'_i$, then the two Safra's trees are the same. Moreover, we have $v_i = v'_i$ if and only if $\bigvee_{j=1}^{n} Q_j \equiv \bigvee_{j=1}^{n} Q'_j$. The decision procedure for formulas equivalence can be guaranteed by satisfiability theorems presented in [12].

Table 2. Algorithm for Deterministic NFG_{inf}.

```
Function DNFG(Q)
/*precondition: $G_{Rabin} = (V_{inf}(Q), E_{inf}(Q), v_0, \Omega)$ is an $NFG_{inf}$ for PPTL formula $Q$. */
/*postcondition: DNFG(Q) outputs a deterministic $NFG_{inf}$ and
  $G'_{Rabin} = (V'_{inf}(Q), E'_{inf}(Q), v'_0, \Omega')$ */
begin function
  $V'_{inf}(Q) = \{Q\}; E'_{inf}(Q) = \emptyset; v'_0 = v_0; E_v = F_v = \emptyset$; /*initialization*/

  while $R \in V'_{inf}(Q)$ and there exists an input $P_i$ do
    foreach node $v \in R$ such that $R \cap F \neq \emptyset$
      do $v' = v \cap F$; $R' = R \cup \{v'\}$; /* create a new node $v'$ such that $v'$ is a son of $v$*/
    foreach node $v$ in $R'$
      do $v = \{P'_i \in V_{inf}(Q) \mid \exists(P, P_i, P'_i) \in E_{inf}(Q), P \in v\}$; /*update $R'$*/
    foreach $v \in R'$ do if $P_i \in v$ such that $P_i \in$ left sibling of $v$ then remove $P_i$ in $v$;
    foreach $v \in R'$ do if $v = \emptyset$ then remove $v$;
    foreach $v \in R'$ do if $u_1, \ldots, u_n$ are all sons of $v$ such that $v = \cup_i\{u_i\}$ $(1 \leq i \leq n)$
                        then remove $u_i$; mark $v$ with !;
    $V'_{inf}(Q) = \{R'\} \cup V'_{inf}(Q)$; $E'_{inf}(Q) = (R, P_i, R') \cup E'_{inf}(Q)$;
  end while
  /*Rabin acceptance components*/
  $E_v = \{R \in V'_{inf}(Q) \mid$ R is Safra tree without node $v\}$;
  $F_v = \{R \in V'_{inf}(Q) \mid$ R is Safra tree with $v$ marked !$\}$;
  return $G'_{Rabin}$;
end function
```

4.3 Product Markov Chains

Definition 7 Let $M = (S, Prob, \iota_{init}, AP, L)$ be a Markov chain M, and for PPTL formula Q, $G_{Rabin} = (V_{inf}(Q), E_{inf}(Q), v_0, \Omega)$ be a deterministic NFG_{inf}, where $\Omega = \{(E_1, F_1), \ldots, (E_k, F_k)\}$. The product $M \otimes G_{Rabin}$ is the Markov chain, which is defined as follows.

$$M \otimes G_{Rabin} = (S \times V_{inf}(Q), Prob', \iota_{init}, \{acc\}, L')$$

where

$$L'(\langle s, Q' \rangle) = \begin{cases} \{acc\} & \text{if for some } F_i, Q' \in F_i, \\ & \text{and } Q' \notin E_j \text{ for all } E_j, \\ & 1 \le i, j \le k \\ \emptyset & \text{otherwise} \end{cases}$$

$$\iota'_{init}(\langle s, Q' \rangle) = \begin{cases} \iota_{init} & \text{if } (Q, L(s), Q') \in E_{inf} \\ 0 & \text{otherwise} \end{cases}$$

and transition probabilities are given by

$$\begin{aligned} & Prob'(\langle s', Q' \rangle, \langle s'', Q'' \rangle) \\ = & \begin{cases} Prob(s', s'') & \text{if } (Q', L(s''), Q'') \in E_{inf} \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

A bottom strongly connected components (BSCCs) in $M \otimes G_{Rabin}$ is accepting if it fulfills the acceptance condition Ω in G_{Rabin}.

For some state $s \in M$, we need to compute the probability for the set of paths starting from s in M for which Q holds, that is, the value of $Pr^M(s \models Q)$. From Definition 7, it can be reduced to computing the probability of accepting runs in the product Markov chain $M \otimes G_{Rabin}$.

Theorem 6 Let M be a finite Markov chain, s a state in M, G_{Rabin} a deterministic NFG_{inf} for formula Q, and let U denote all the accepting BSCCs in $M \otimes G_{Rabin}$. Then, we have

$$Pr^M(s \models G_{Rabin}) = Pr^{M \otimes G_{Rabin}}(\langle s, Q' \rangle \models \Diamond U)$$

where $(Q, L(s), Q') \in E_{inf}$.

Corollary 7 All the ω-regular properties specified by PPTL are measurable.

Example 3 We now consider the example in Figure 1. Let M denote Markov chain in Figure 1(b). The probability that *sequential property* p **;** q holds in Markov chain M can be computed as follows.

First, by the two algorithms above, deterministic NFG_{inf} with Rabin condition for p **;** q is constructed as in Figure 1(a), where the Rabin acceptance condition is $\Omega = (v_1, v_2)$. Further, the product of the Markov chain and NFG_{inf} for formula p **;** q is given in Figure 6.

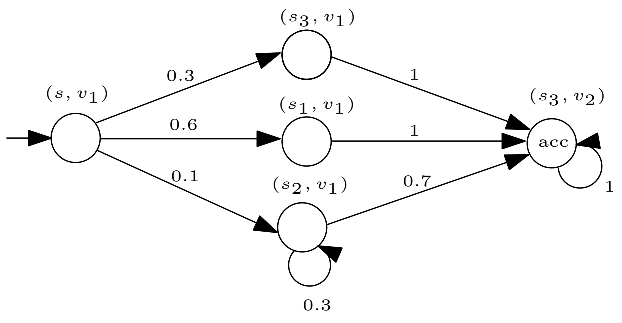


Fig. 6. The Product of Markov chain and NFG_{inf} in Figure 1.

From Figure 6, we can see that state (s_3, v_2) is the unique accepting BSCC. Therefore, we have

$$\begin{aligned} & Pr^M(s \models G_{Rabin}) \\ = & Pr^{M \otimes G_{Rabin}}((s, v_1) \models \Diamond(s_3, v_3)) \\ = & 1 \end{aligned}$$

That is, sequential property p **;** q is satisfied almost surely by the Markov chain M in Figure 1(b).

5 Conclusions

This paper presents an approach for probabilistic model checking based on PPTL. Both propositional LTL and PPTL can specify linear-time properties. However, unlike probabilistic model checking on propositional LTL, our approach uses NFGs, not Buchi automata, to characterize models of logic formulas. NFGs possess some merits that are more suitable to be employed in model checking for interval-based temporal logics.

Recently, some promising formal verification techniques based on NFGs have been developed, such as [13,14]. In the near future, we will extend the existing model checker for PPTL with probability, and according to the algorithms proposed in this paper, to verify the regular safety properties in probabilistic systems.

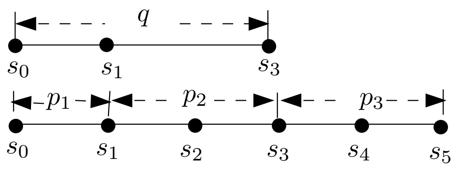
Acknowledgement

Thanks to Huimin Lin and Joost-Pieter Katoen for their helpful suggestions.

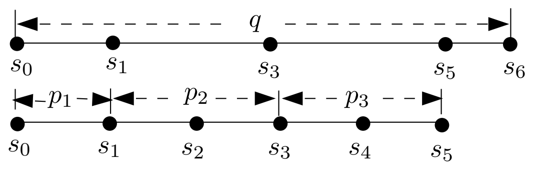
References

1. H. Hansson and B. Jonsson. (1994), A Logic for Reasoning about Time and Reliability. Formal Aspects of Computing. Vol. 6, pages 102-111.

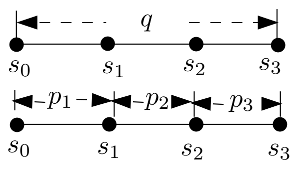
2. A. Aziz, K. Sanwal, V. Singhal and R. K. Brayton. (2000), Model Checking Continous Time Markov Chains. ACM Trans. Comput. Log. Vol. 1(1): 162-170.
3. B. Moszkowski. (1983), Reasoning about digitial circuits. PhD Thesis. Stanford University. TRSTAN-CS-83-970.
4. B.Moszkowski. (1986), Executing temporal logic programs. Cambridge University Press.
5. M. Z. Kwiatkowska, G. Norman and D. Parker. (2004), Probabilistic Symbolic Model Checking with PRISM: a hybrid approach. *STTT*. Vol. 6(2): 128-142.
6. C. Baier, J. P. Katoen. (2008), Principles of Model Checking. The MIT Press.
7. J. -P. Katoen, M. Khattri and I. S. Zapreev. (2005), A Markov Reward Model Checker. In *QEST*, pages 243-244.
8. J. G. Kemeny and J. L. Snell. (1960), Finite Markov Chains. Van Nostrad, Princeton.
9. P. L. Wolper. (1983), Temporal logic can be more expressive. *Information and Control*, vol.56, pages 72-99. 1983.
10. Z.Duan, X.Yang and M.Koutny. (2008), Framed Temporal Logic Programming. *Science of Computer Programming*, Volume 70(1), pages 31-61, Elsevier North-Holland.
11. X. Yang and Z. Duan. (2008) Operational Semantics of Framed Tempura. *Journal of Logic and Algebraic Programming*, Vol.78(1):22-51, Elsevier North-Holland.
12. Z. Duan, C. Tian, L. Zhang. (2008), A Decision Procedure for Propositional Projection Temporal Logic with Infinite Models. *Acta Informatic*, Springer-Verlag, 45, 43-78.
13. C.Tian and Z. Duan. (2010), Making Abstraction Refinement Efficient in Model Checking CoRR abs/1007.3569.
14. C. Tian, Z. Duan. (2007), Model Checking Propositional Projection Temporal Logic Based on SPIN. ICFEM pages: 246-265.
15. E. Grädel, W. Thomas, T. Wilke (Eds.). (2002), Automata Logics, and Infinite Games: A Guide to Current Research. Springer-Verlag Berlin Heidelberg.



(a): q terminates before p_3


q
s_0 s_1 s_3 s_5 s_6
p_1 p_2 p_3
s_0 s_1 s_2 s_3 s_4 s_5

(b): p_3 terminates before q



(c): q and p_3 terminate at the same poi